The systematic design of systolic arrays

Patrice Quinton

INRIA

CENTRE DE RENNES

IRISA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél. : (3) 954 90 20

Rapports de Recherche

N° 216

THE SYSTEMATIC DESIGN OF SYSTOLIC ARRAYS

Patrice QUINTON

Juillet 1983

THE SYSTEMATIC DESIGN OF SYSTOLIC ARRAYS

Patrice QUINTON

RESUME :

Les architectures systoliques représentent une classe de machines parallèles spécialisées particulièrement intéressantes compte tenu de leur régularité et des performances qu'elles permettent d'atteindre. On connaît de très nombreux exemples d'algorithmes qui peuvent être résolus sur de telles architectures. Toutefois, chacune des solutions proposée a été obtenue par des moyens heuristiques. Nous proposons une méthode systématique permettant de construire des solutions systoliques pour la classe des algorithmes qui peuvent être exprimées au moyen d'équations récurrentes uniformes sur un domaine convexe D de Z^n. La méthode consiste à

- construire une fonction de temps affine t de D dans N compatible avec l'ordre induit sur D par le système d'équations uniformes ; cette fonction indique à quel instant chaque calcul doit être effectué ;

- définir une fonction d'allocation a de D dans un ensemble fini a(D) de Z^n, de telle sorte que les calculs effectués au même instant soient alloués à deux points différents de a(D).

Les conditions que doivent satisfaire t et a permettent de les obtenir d'une façon constructive. La méthode permet de retrouver de nombreuses architectures systoliques connues et d'en dériver de nouvelles, comme on le montre sur les exemples classiques du produit de convolution et du produit de matrice.

.../...

SUMMARY :

Systolic arrays are a type of special-purpose parallel devices that are particularly attractive, since they are very regular and may be used for solving in real time computation consuming algorithms. A number of systolic arrays have been designed for solving a large class of useful algorithms. However, these designs have been obtained in a heuristic manner. We propose here a systematic method for the design of systolic arrays. This method may be used for algorithms that can be expressed as a set of uniform recurrent equations (URE) over a domain D of Z^n. The methods consists in :

- finding an affine timing-function t from D to N which is compatible with the dependences between computations that are introduced by the URE ; this function determines when a calculation may be made ;

- defining an allocation function a from D to a finite subset a(D) of Z^n, in such a way that computations that may be executed simultaneously are mapped on different points. a defines the structure of the systolic array.

This method has the main advantage to be constructive. It is shown it may be applied to find well-known designs and also new ones in the case of the convolution product and matrix product.

THE SYSTEMATIC DESIGN OF SYSTOLIC ARRAYS

Patrice QUINTON
IRISA, Campus de Beaulieu,
35042 RENNES-CEDEX

1. INTRODUCTION

Progress in VLSI technology allows us to conceive of devices that are made out of a number of processors and thus offers us exceptional opportunities to develop parallel computations, for both special-purpose and general-purpose devices. Among the several approaches to parallel organization that can take advantage of these new facilities, the systolic array concept is particularly interesting. As characterized by KUNG [2], a systolic array is a parallel device, made out of a few simple cell types, regularly and locally connected. Data circulate through these cells in a very regular fashion and interact where they meet, giving results that are pumped out of the cells. Systolic arrays enjoy some nice properties:

- for compute-bound problems, they allow a maximum of computation for a minimum of I/O operations;

- they need no complicated control, and may be implemented as synchronous systems, avoiding the well-known overhead problem encountered in other parallel organizations like MIMD machines;

- their regular structure makes them particularly well-suited for VLSI implementation, since the replication factor of the design may be kept high. Also, local communication greatly simplifies synchronization and control problems.

A number of such devices has been proposed for solving a large class of useful algorithms including signal processing, numerical analysis, database management, etc... However, every systolic array that has been proposed seems to have been designed in a heuristic manner.

The purpose of this paper is to propose a systematic way for building such devices. The method proposed here is based on the possibility of expressing a problem as a set of uniform recurrence equations [1] over a set of integer coordinates points of R^n. When this is the case, it is possible under particular conditions to order the computations in such a way that the entire domain may be mapped into a systolic array.

The following paper is organized in six parts. In the second part, the principle of the method is described, somewhat informally, starting from the well-known example of the convolution algorithm, for which a number of designs exist (see for example KUNG [2]). In the third part, the problem is stated more formally, and we give a necessary and sufficient condition under which a set of uniform recurrence equations may be scheduled, using a timing-function that is called **quasi-affine**. Moreover, an automatic method for finding timing-functions is described for the case when **D** is a convex set of $\mathbf{R}^n$.

Part four deals with the next step of the method. Once a timing- function is obtained, one may project the computation domain into a finite machine, using a linear allocation function. Part five details how to specify a systolic array from the equations, the domain of computation, the timing- and allocation-functions. Finally, part six describes how the method may be used for convolution, and matrix product.

2. INFORMAL EXPLANATION OF THE METHOD: THE CONVOLUTION PRODUCT

Given a sequence $x_0, x_1, x_2, \dots, x_i, \dots$ and coefficients $w_0, w_1, \dots, w_K$, the convolution algorithm consists in computing the sequence $y_0, y_1, \dots, y_i, \dots$ where y_i is given by the equation

(1) $y_i = \sum w_k \, x_{i-k}$

Equation (1) may be rewritten as

$$y_{i,k} = y_{i,k-1} + w_k \, x_{i-k} \qquad 0 \leqslant i \quad 0 \leqslant k \leqslant K$$

(2) $$y_{i,-1} = 0 \qquad 0 \leqslant i$$

where $y_{i,k}$ $(0 \leqslant k \leqslant K)$ are the partial accumulated values for y_i. We are interested here in finding systolic arrays that compute (2).

For any integer coordinate point i,k lying in the domain $\mathbf{D} = \{0 \leqslant i \; ; \; 0 \leqslant k \leqslant K\}$, we have to compute a function $y_{out} = y_{in} + w_{in} \, x_{in}$, which will deliver $y_{i,k}$ provided y_{in}, w_{in}, and x_{in} are given correct values $y_{i,k-1}$, x_{i-k}, w_k.

It can be easily seen that computation at point i,k cannot take place before computation at point i,k-1 has been achieved, since $y_{in} = y_{i,k-1}$ is computed by node i,k-1.

According to this observation, it would be very simple to order computations of (2) in such a way that these dependence constraints be fulfilled: for example, a schedule that would fit these constraints is to compute $y_{i,k}$ at time t = k. But such a computation scheme has the undesirable property that values w_k and x_{i-k} must be input from memory several times, requiring a high bandwidth between memory and computation units.

An alternative to this first natural schedule is to make w_k and x_{i-k} circulate from computation node to computation node. For example, w_k, which is necessary for computing $y_{i,k}$, may be obtained as a byproduct of the computation of $y_{i-1,k}$. In the same manner, one can suppose that x_{i-k}, which is needed for computing $y_{i,k}$, is given as a byproduct of computation $y_{i-1,k-1}$. Note that other similar schemes are possible, and the first step of our method is to define precisely where values necessary for each computation are to be taken.

Let us assume for example that input value w_k of node i,k is provided by node i-1,k , and x_{i-k} by node i-1,k-1 . Such a scheme may be formally expressed as the following system of equations:

$\forall i : 0 \leq i, \forall k : 0 \leq k \leq K$

$$(3) \begin{cases} y(i,k) = y(i,k-1) + w(i-1,k) \times x(i-1,k-1) \\ w(i,k) = w(i-1,k) \\ x(i,k) = x(i-1,k-1) \end{cases}$$

Such a system of recurrent equations is said to be **uniform**, since computation at point i,k depends only on values computed at points that are obtained by a translation which does not depend on i or k (see [1]). Such a system may be represented by a graph such as that of Fig. 1. The nodes of this graph represent the computations to be achieved and the edges represent values that are to be transmitted from one node to another. Such a graph may be more concisely abstracted as a domain D of Z^2 and a set of dependence vectors $\Theta = \{\theta_w, \theta_y, \theta_x\}$ that defines where node i,k has to take its input values, where

$\theta_w = (-1,0)$

$\theta_y = (0,-1)$

$\theta_x = (-1,-1)$

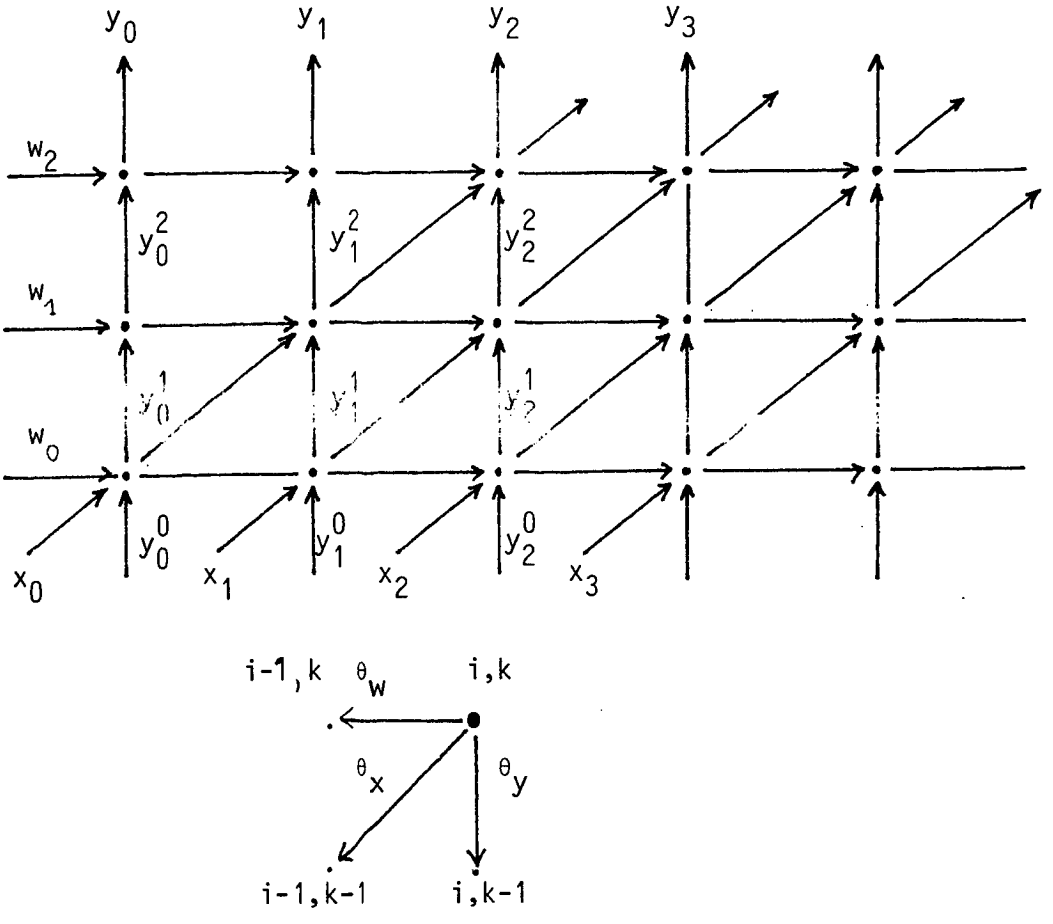


Fig. 1: Dependence graph for the convolution product (K=2)

The next step of the method is to define a schedule for the computation by means of a **timing-function** . A timing-function is a mapping t from **D** to **N** such that if computation at point x of **D** depends on those at point y, then t(x) > t(y). Such a function does not always exist. This will be the subject, of the third part to define conditions that the θ_i's and **D** must fulfil in order for t(x) to exist. For the moment, let us note only that a very simple linear timing-function exists in the case of the above example, namely (see Fig. 2)

(4) t(i,k) = i+k

According to this timing-function, it is possible to solve (3) by successively executing computations for points x $\in$ **D** whose timing value t(x) is 0,1,...,n,...

The last step of our method is to define an architecture that supports these calculations. This is obtained by defining an **allocation function a** which maps **D** into a finite set. Here also, a convenient way to derive simple solutions is to define the allocation function as a mapping from **D** to a finite domain of **Z** by means of a linear function. For our example, one may choose

(5) a(i,k) = k

A very simple condition that such a function must satisfy is that the lines it defines are not parallel to the timing lines; when this condition is fulfilled, one is sure that any processor has at most one computation to execute at a given instant. The functions t and a respectively given by (4) and (5) define completely a well-known systolic device. The whole domain **D** is mapped into 3 processors p_0, p_1, and p_2. Communications among the processors are completely defined, since dependence vectors θ_x, θ_y, θ_w define where a processor gets or sends values and t(i,k) defines what a processor has to compute. Notice also that t defines the speed of the flows of data through the cells; as an example, a carefull examination of Fig. 1 reveals that the x_i's progress from cell to cell every two other time. Fig. 3 shows the obtained architecture.

In summary, the method we informally described consists of three steps:

(1) Find some uniform recurrence equation system (URE in the following) that solves the given problem,

(2) Find a timing-function for the URE,

(3) Find an allocation function.

In the following, we will focus on problems (2) and (3). Although very important, step (1) will not be treated here.

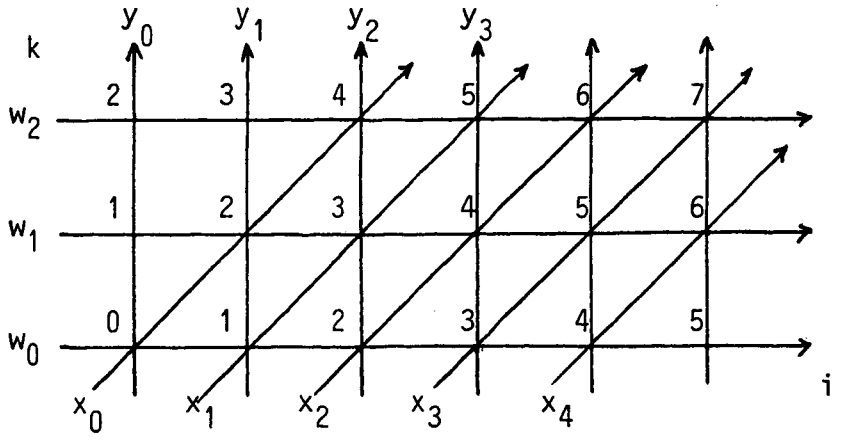


Fig. 2: A possible timing-function for the dependence graph of Fig. 1, which is defined by t(i,k) = i+k .

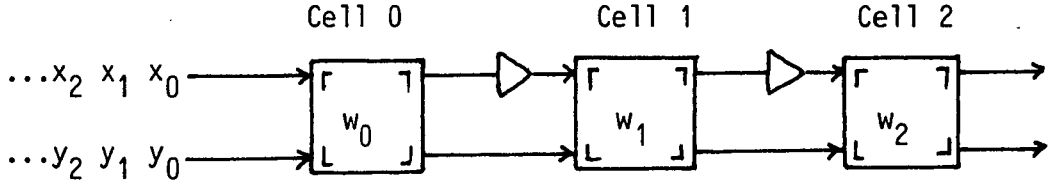


Fig. 3: The systolic array which is obtained from Fig.2 with a(i,k) = k.
A small triangle on a communication link indicates that data has to stage one unit of time before entering the next cell

3. DEPENDENCE GRAPH AND TIMING-FUNCTION

As we have seen in part 2, one major step of the method is to find some timing-function for the computation graph. This part deals with that problem and shows that, under a few simple conditions, we can find timing-functions for a given URE. The first paragraph deals with the general case; the second one concerns the case of convex polyhedral domains of Z^n.

3.1 General case

Let R^n be the n-dimensional real-space and Z^n be the subset of integer coordinate points of R^n. Let $D \subset Z^n$ be a **domain** of computation. For every point z of D, we have to compute a system of equations E(z) of the following form:

$$E(z)\begin{cases} u_1(z) = f(u_1(z+\theta_1), u_2(z+\theta_2), \ldots, u_p(z+\theta_p)) \\ u_2(z) = u_2(z+\theta_2) \\ \quad . \\ \quad . \\ u_p(z) = u_p(z+\theta_p) \end{cases}$$

Such a system of equations is said to be uniform if vectors θ_i (called **dependence vectors**) do not depend on z. Notice that for some points z of D, the calculation of E(z) involves some values $u_i(y)$ where y lies outside of D. Such values are assumed to be known. In the following, the pair (D,Θ) will be called a **dependence graph** (*).

Computation at point x is said to **depend strictly** on computation at point y (or shortly x depends strictly on y) if

(6) $\exists \theta_i \quad y = x+\theta_i$

x **depends** on y if there exists a finite sequence $x_1, x_2, \ldots, x_k$ such that

$x_1 = x$

(7) $x_k = y$

x_i depends strictly on x_{i+1} $\forall$ i: $1 \leq i < k$.

(*) The system we consider here is a simplified version of uniform recurrence equations defined by KARP et al. [1].

It is obvious that (7) is equivalent to

(8) $\exists\ (\alpha_1,\ldots,\alpha_p) \in \mathbf{N}^p - \{0\}$ such that $y = x + (\sum_{i=1}^{p} \alpha_i\theta_i)$

In the following, we suppose for the sake of simplicity that every computation takes one unit of time. We are interested in finding possible timing-functions for **D**. A timing-function is a mapping t from **D** to **N** such that

(9) x depends on y $\Rightarrow$ t(x)>t(y)

We shall consider here a particular class of timing functions which we call **quasi-affine** timing-functions (QATF). Such a function has the form

(10) $t(x) = \lfloor {}^t\lambda x - \alpha \rfloor \quad \lambda \in \mathbf{R}^n$ and $\alpha \in \mathbf{R}$.

where $\lfloor u \rfloor$ means the greatest integer lower or equal to u. The class of **affine timing-function** (ATF in the following) is a particular subset of the QATF set which verify $\lambda \in \mathbf{Z}^n$ and $\alpha \in \mathbf{Z}$. In the following, we shall denote a QATF as a pair (λ,α).

The following theorem gives a sufficient condition under which a dependence graph has a QATF. Moreover, it is shown that this condition is necessary when the timing-function is constrained to be an ATF.

Theorem 1

$t=(\lambda,\alpha)$ is a QATF for $(\mathbf{D},\Theta)$ if

(i) ${}^t\lambda x - \alpha \geqslant 0 \quad \forall x \in \mathbf{D}$

(ii) $-{}^t\lambda\theta \geqslant 1 \quad \forall\theta \in \Theta$

Moreover, if t is an ATF, then (i) and (ii) are necessary.

Before presenting the proof of this theorem, let us give a geometric interpretation of it, as illustrated by Fig.4 in the particular case of $\mathbf{R}^2$. The theorem simply means that there exists a line separating $\mathbf{R}^2$ in two half-planes, one of them containing **D** and the other one containing the dependence vectors Θ_i when reported from a unique point A of this line.

Proof

Suppose there exists λ and α such that (i) and (ii) hold. We shall show that $t=(\lambda,\alpha)$ is a QATF for (D,Θ). First of all, (i) implies obviously that $t(x) \geqslant 0$ over D. On the other hand, let x and y be two points of D such that x depends on y. Equality (8) implies that

$$x = y - \sum_{i=1}^{p} \alpha_i \theta_i$$

where ${}^t(\alpha_1, \ldots, \alpha_p)$ is a non-negative vector of Z^p.

We may write

$$t(x) = {}^t\lambda x - \alpha = {}^t\lambda(y - \sum_{i=1}^{p} \alpha_i\theta_i) - \alpha = {}^t\lambda y - \alpha - \sum_{i=1}^{p} \alpha_i {}^t\lambda\theta_i \geqslant t(y) - \sum_{i=1}^{p} \alpha_i {}^t\lambda\theta_i$$

But (ii) implies that $-{}^t\lambda\theta_i \geqslant 1$ for every i, and since one of the α_i at least is greater or equal than 1, we may conclude that $t(x) \geqslant t(y)+1$.

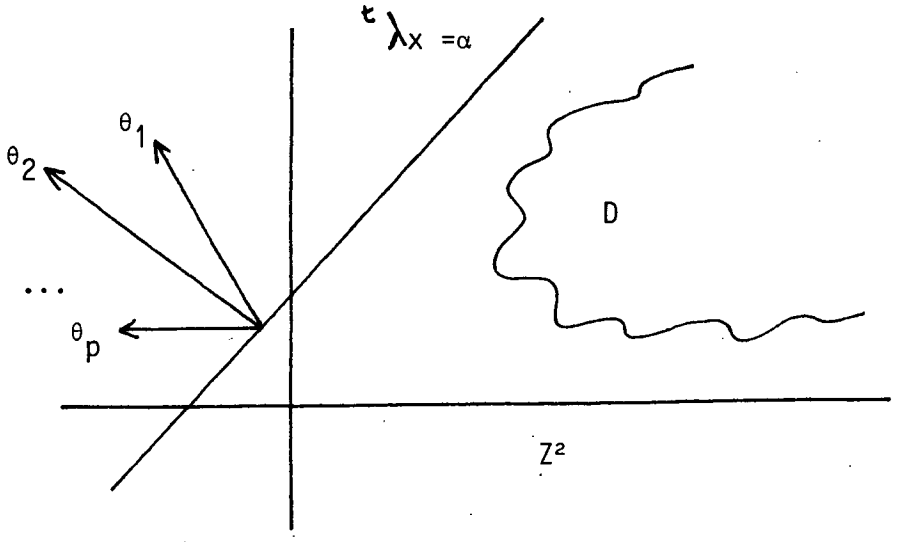


Fig. 4: Illustration of theorem 1.

Suppose now that t is an ATF, i.e. $\lambda \in Z^n$ and $\alpha \in Z$. Clearly, (i) is true since t(x) is positive over **D**. On the other hand, $\forall \theta \in \Theta$, we have

$$t(x) - t(x+\theta) \geqslant 1$$

or

$$^t\lambda x - \alpha - (^t\lambda(x+\theta) - \alpha) \geqslant 1$$

or

$$t(x) - (t(x) + {}^t\lambda\theta) \geqslant 1$$

which implies $-{}^t\lambda\theta \geqslant 1$. As a consequence, (ii) is also true. **QED**

In the case of QATF, the above theorem gives only a sufficient condition for finding timing-functions. The compelling reason is that when the domain D is bounded, there exist infinitely many QATF's that have the same value over D, but differ at least at some point of $Z^n - D$. However, as we have seen on the example of the convolution product, what we want to build is not a systolic array for one singular domain, but more generally, a class of solutions for a class of problems. As an example, it is clear that the solution we gave in part 2 is extendable very naturally to any value of K. The following deals with this particular problem.

Consider a **family of dependence graphs** $F = \{(D_i,\Theta)\}_{i \in I}$ where $I \subset \mathbf{N}$. We shall now look for a **family of QATF's** $\{t_i=(\lambda,\alpha_i)\}_{i \in I}$ where $\forall i$, t_i is a QATF for (D_i,Θ).

Notice that Θ is constant over the whole family of dependence graphs, and λ is constant for all of the QATF's. We say that F is **extendable** if, for every $\theta \in \Theta$ and any integer q of **N**, there exists a domain D_i of the family that contains a sequence $\{x, x+\theta, \ldots, x+q\theta\}$ for some point x of D_i. As an example, the first orthant $\{x \geqslant 0 \mid x \in Z^n\}$ is extendable for any finite set Θ of vectors of Z^n. In the same way, the family $F = \{(D_K,\Theta)\}_{K \in \mathbf{N}}$ where Θ is a finite set of vectors of Z^2 and

$$D_K = \{(i,k) \mid 0 \leqslant i \text{ and } 0 \leqslant k \leqslant K\}$$

is extendable, although any finite subfamily of F is not, as soon as Θ contains a vector whose second component is non-null. We have then the following theorem:

Theorem 2

Let $F = \{(D_i,\Theta)\}_{i\in I}$ be an extendable family of dependence graphs. Then $\{t_i=(\lambda,\alpha_i)\}_{i\in I}$ is a family of QATF's for F iff

(i) $\forall_i, \forall_{x\in D_i}, {}^t\lambda x-\alpha_i \geqslant 0$

(ii) $\forall\theta\in\Theta, \ -{}^t\lambda\theta \geqslant 1$

Proof: we need only to prove that (i) and (ii) are necessary since Theorem 1 implies that (i) and (ii) are sufficient. Clearly, (i) is true, otherwise t_i would not be a timing-function for at least one index i. Let us now prove that (ii) is true. Notice first that $\forall\theta, \ -{}^t\lambda\theta>0$, because, since F is extendable, there exist D_i and $x\in D_i$ such that $x+\theta\in D_i$. Thus

$$t_i(x) - t_i(x+\theta)\geqslant 1$$

or

$$\lfloor {}^t\lambda x-\alpha_i \rfloor - \lfloor {}^t\lambda x-\alpha_i + {}^t\lambda\theta \rfloor \geqslant 1$$

which implies $-{}^t\lambda\theta > 0$.

The remaining of the proof deals with showing that every rational which is greater than or equal to $-{}^t\lambda\theta$ is also greater than or equal to 1. This will suffice to prove that $-{}^t\lambda\theta \geqslant 1$, since

$$-{}^t\lambda\theta = \inf\{\alpha\in\mathbf{Q} \mid \alpha\geqslant -{}^t\lambda\theta\}$$

Let p/q be a rational number such that $-{}^t\lambda\theta\leqslant p/q$. There exist D_i and $x\in D_i$ such that $\{x, x+\theta, \ldots, x+q\theta\} \in D_i$. We have then

$$t_i(x) \geqslant t_i(x+\theta)+1 \geqslant \ldots \geqslant t_i(x+q\theta) + q$$

but

$$t_i(x+q\theta) = \lfloor {}^t\lambda x-\alpha_i + q\,{}^t\lambda\theta \rfloor \geqslant \lfloor {}^t\lambda x - \alpha_i - p \rfloor = t_i(x) - p$$

since $-{}^t\lambda\theta \leqslant p/q$ and $p/q \geqslant 1$. We have thus

$$t_i(x)- q \geqslant t_i(x+q\theta) \geqslant t_i(x)-p$$

which implies that $p \geqslant q$, or $p/q \geqslant 1$. **QED**

This theorem, although very general, does not give any constructive way to find a timing-function. Fortunately, in most cases, **D** is a convex set which is described by a set of linear inequalities. The following section gives a constructive approach for this case.

3.2 The case when D is a convex set

In most cases, equations of a problem involve indexes that lie in a convex domain of $\mathbf{R}^n$. As we shall see here, it is possible to derive automatically QATF's for this particular case, using elementary convex analysis (see [7]).

3.2.1 Definitions

Suppose that **D** is defined by a set of inequalities, as is the case for the convolution product. **D** is then the subset of integer-coordinate points of a convex polyhedral domain (CPD) of $\mathbf{R}^n$ which will be named $\underline{D}$ in the following. $\underline{D}$ may thus be defined by a set of inequalities, i.e.

(11) $\underline{D} = \{Ax \leqslant d \mid x \in \mathbf{R}^n\}$

where A is a $q \times n$ matrix and d a $q \times 1$ vector over Z.

A few definitions and basic properties are in order for the clarity of the following development. Let $x_1,\ldots,x_n$ be points of $\mathbf{R}^n$. We say that $\sum_{i=1}^{n}\mu_i x_i$ is a **positive combination** of $x_1,\ldots,x_n$ if μ_i are non-negative real numbers. $\sum_{i=1}^{n}\alpha_i x_i$ is a **convex combination** of $x_1,\ldots,x_n$ if it is a positive combination of $x_1,\ldots,x_n$ and $\sum_{i=1}^{n}\alpha_i = 1$. s is a vertex of $\underline{D}$ if s cannot be expressed as a convex combination of two different points of $\underline{D}$. r is a **ray** of $\underline{D}$ if

$$\forall x \in \underline{D}, \forall \mu \in \mathbf{R}^+ \quad x + \mu r \in \underline{D}$$

A ray r of $\underline{D}$ is said to be **extremal** if r cannot be expressed as a positive combination of other rays of $\underline{D}$. l is a **line** of $\underline{D}$ if

$$\forall x \in \underline{D}, \forall \mu \in \mathbf{R} \quad x + \mu l \in \underline{D}$$

If $\underline{D}$ contains a line, it is said to be a cylinder. In the following, we shall consider only CPD that are not cylinders. In such a case, the set S of vertices of $\underline{D}$ is unique, and the set R of extremal rays of $\underline{D}$ is unique up to a non-zero scalar multiple. $\underline{D}$ may then be defined as the subset of points x of $\mathbf{R}^n$ such that $x = y + z$, where y is a convex combination of vertices of S and z is a positive combination of rays of R. The pair (S,R) will be called a **system of generators** of $\underline{D}$. Notice that $S \subset \mathbf{Z}^n$ since $\underline{D}$ is the convex hull of D and thus has integer coordinate vertices.

There is, of course a strong relationship between the two above characterizations of a CPD. Let J be a subset of $I=\{1,...,q\}$, the row index set of matrices A and d. Given a $q \times m$ matrix M, let us denote as M^J the submatrix of M consisting of the rows of M whose index is in J. Then s is a vertex of $\underline{D}$ iff there exists an index set J of size n such that:

$$A^J s = d^J$$
$$(12)\quad A^{I-J} s \leqslant d^{I-J}$$
$$A^J \text{ is regular}$$

Also, r is an extremal ray of $\underline{D}$ iff there exists an index set J of size n-1 such that:

$$Ar \leqslant 0$$
$$(13)\quad A^J = 0$$
$$A^J \text{ is rank-n}$$

A CPD is said to be n-dimensional if the smallest affine space generated by $\underline{D}$ is n-dimensional. In the following, we shall always consider CPD $\underline{D}$ which are n-dimensional. Of course, in this case, A is a rank-n matrix.

Example.

Fig.5 shows a CPD of R^2 which may be represented as

$$\underline{D} = \{x \mid Ax \leqslant d\}$$

where

$$A = \begin{bmatrix} 1 & -1 \\ 1 & 1 \\ -\frac{1}{2} & \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \qquad d = \begin{bmatrix} -1 \\ -1 \\ -2 \\ -1 \end{bmatrix}$$

The system of generators of $\underline{D}$ is the pair (S,R) where

$S=\{(4,1),(2,1),(2,3)\}$

$R=\{(1,1),(2,1)\}$

End of example

3.2.2 Timing-functions for convex polyhedral domains

Let us return to our initial problem. Given a dependence graph (D,Θ), we want to find the set of QATF's. The following theorem gives a characterization for it:

Theorem 3

$t=(\lambda,\alpha)$ is a QATF for (D,Θ) if

(i) $-{}^t\lambda\theta \geqslant 1 \quad \forall\theta\in\Theta$

(ii) ${}^t\lambda r \geqslant 0 \quad \forall r\in R$

(iii) $\alpha \leqslant {}^t\lambda s \quad \forall s\in S$

Moreover, if t is an ATF, then (i), (ii), and (iii) are necessary.

Proof

Every $x\in\underline{D}$ may be written

$$x=\Sigma\alpha_i s_i + \Sigma\mu_j r_j \quad s_i\in S,\ r_i\in R$$

thus

$${}^t\lambda x-\alpha = \Sigma\alpha_i\,{}^t\lambda s_i + \Sigma\mu_j\,{}^t\lambda r_j-\alpha$$

But from (iii), we get

$$\Sigma\alpha_i\,{}^t\lambda s_i \geqslant \Sigma\alpha_i\alpha = \alpha$$

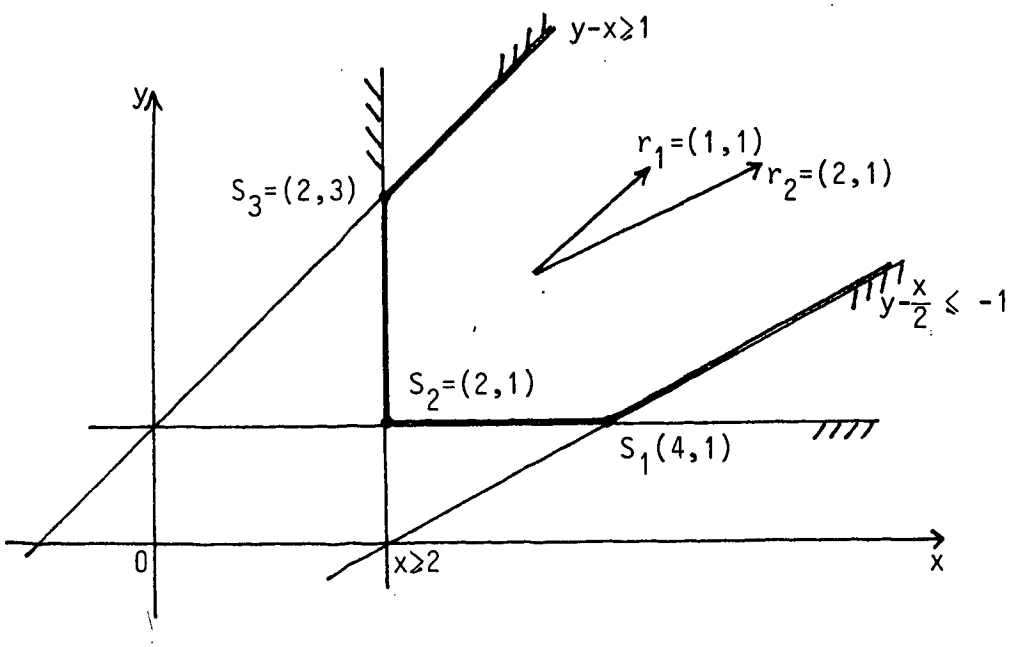


Fig. 5: Example of a Convex Polyhedral Domain of R^2.

Thus

$$^t\lambda x-\alpha \geqslant \Sigma\mu_j\, ^t\lambda r_j$$

Since λ meets condition (ii), we can conclude that

(iv) $$\left[\,^t\lambda x-\alpha\right]_j \geqslant 0 \quad \forall x\in \underline{D}$$

As a consequence of (iv) and (i), theorem 1 holds. Consequently, (λ,α) is a QATF.

Now suppose t is an ATF. (i) results immediately from theorem 1. (iii) is obvious, since otherwise t(x) would be negative, at least at one vertex of $\underline{D}$ and would not be a timing-function. Finally, suppose (ii) is not true, i.e. for some extremal ray r of R, $^t\lambda r<0$. By definition of a ray

$$\forall\mu>0,\ \forall x\in\underline{D}:\ x+\mu r\in\underline{D}$$

It is then clear that for some x and for some μ, $^t\lambda(x+\mu r)-\alpha<0$ which is not consistent with the definition of a timing-function. QED

Example

Consider again the example of convolution. $\underline{D}$ is the set $\{(i,k)\in\mathbb{R}^2 \mid 0\leqslant i$ and $0\leqslant k\leqslant K\}$. The unique system of generators of $\underline{D}$ is $S=\{(0,0),(0,K)\}$ and $R=\{(1,0)\}$. Applying theorem 3 gives the convex set Λ described by Fig. 6.

$$\Lambda = \{(\lambda_1,\lambda_2) \mid \lambda_1\geqslant 1\ ,\ \lambda_2\geqslant 1\ ,\ \lambda_1+\lambda_2\geqslant 1\ ,\ \lambda_1\geqslant 0\}$$

Λ itself is a CPD of $\mathbb{R}^2$ which has a single vertex $\sigma=(1,1)$ and two extremal rays $\rho_1=(1,0)$ and $\rho_2=(0,1)$. The timing function we defined in the second part of our paper is $t(i,k) = i+k = {}^t\sigma.{}^t(i,k) - \alpha$ where $\alpha = 0$ meets condition (ii).

End of example.

Notice that theorem 3 holds whether $\underline{D}$ is finite or not; if D is finite, condition (ii) obviously holds since $R=\emptyset$.

In the same way as theorem 2 extends theorem 1, we can extend theorem 3 to the case of families of convex dependence graphs:

Theorem 4

Let $F = \{(D_i,\Theta)\}_{i\in I}$ be an extendable family of dependence graphs. Suppose moreover that every D_i has the same set R of extremal rays. Then $\{t_i=(\lambda,\alpha_i)\}_{i\in I}$ is a family of QATF for F if and only if

(i) $-{}^t\lambda\theta \geqslant 1 \quad \forall\theta\in\Theta$

(ii) ${}^t\lambda r \geqslant 0 \quad \forall r\in R$

(iii) $\alpha_i \leqslant {}^t\lambda s_i \quad \forall s\in S$

Proof: results immediately of theorems 2 and 3.

Theorem 4 gives us a constructive means for finding every QATF of a given family of dependence graphs: (i) and (ii) define a CPD of $\mathbf{R}^n$. Algorithms for finding the vertices and rays of a CPD exist (see [4] for example), and it is thus easy to find automatically possible values of λ and α.

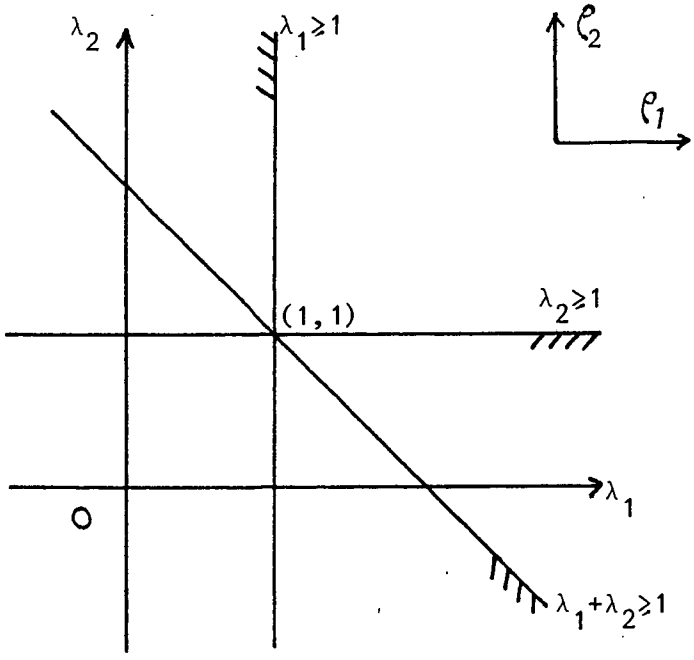


Fig. 6: The convex domain Λ for the convolution product example

4. ALLOCATION FUNCTIONS

As explained in part 2, the third step of our method is to map the computation graph on a finite network of processors. Here also, we shall restrict ourselves to a particular class of mappings that we shall call **linear allocation functions** (LAF). Let n be the dimension of the domain D. Suppose $t=(\lambda,\alpha)$ is a QATF for (D,Θ). An allocation function is a mapping a from D to a finite subset of Z^m, where m is the dimension of the systolic array to be obtained. In the following, m will be taken as either equal to n-1 or to n, depending on t being an ATF or a strict QATF (this point will be made clear later on). However, it must be pointed out that the dimension of the systolic array may be chosen arbitrarily provided the allocation function ensures that the following condition is fullfilled

(14) $\forall_{x,y} \in D \quad a(x)=a(y) \Rightarrow t(x) \neq t(y)$

This condition ensures that the allocation function is conflict-free relative to the timing function t.

An allocation function a from D to Z^m is said to be **linear** if a is a linear mapping from Z^n to Z^m. a is said to be **quasi-linear** (QLAF) if there exists a mapping a' from Z^n to Z^m and a vector P of Z^m such that

(15) $a(x) = a'(x) \bmod P$

We shall now distinguish two cases, namely when t is an ATF, and t is a strict QATF with rational coefficients.

4.1. t is an ATF

Let $t=(\lambda,\alpha(\lambda))$ be such a function. The idea is simply to project the dependence graph along a direction which is not parallel to the timing hyperplanes. We are thus ensured that condition (14) is fullfilled, since every projection line has at most one intersection point with a given timing hyperplane (see Fig.7). Depending on the domain D and on the projection direction that is chosen, one can obtain a finite machine or not. First of all, it should be clear that the domain D and the timing-function must be such that the number of points to be computed at a given time is bounded, i.e.

(16) $\exists M \in \mathbf{N}, \forall n \in \mathbf{N}, \mathrm{card}\{x \mid x \in D \text{ and } t(x)=n\} \leq M$

Such a condition holds clearly when D is finite. But in some other cases (for example, the convolution product), one can map an infinite domain D onto a finite machine. In the following, we shall suppose that D has at most one ray, say r.

Let ξ be a non-zero vector of Z^n. In the following, we shall suppose that ξ and the ray r of D are the smallest integer-coordinate vector in the direction they define (ξ and r are defined uniquely up to a non-zero scalar multiple). If this is the case, any line $L = \{x+\mu v \mid \mu \in \mathbf{R}\}$ (where v is ξ or r) is such that

$$Z^n \cap L = \{x+kv \mid k \in Z\}$$

Since ξ is non-zero, one of its coordinates is non-zero at least. For commodity, suppose $\xi_n \neq 0$. Let us denote e_i ($1 \leqslant i \leqslant n$) the canonical basis vectors of $\mathbf{R}^n$. It is clear that $\{e_1, \ldots, e_{n-1}, \xi\}$ is a basis of $\mathbf{R}^n$. Let $x = {}^t(x_1, \ldots, x_n)$ be a point of $\mathbf{R}^n$, and let $y = {}^t(y_1, \ldots, y_n)$ be the decomposition of x over this new basis. A simple calculation shows that the y_i satisfy the following relations:

$$(17) \quad \begin{cases} y_i = x_i - x_n (\xi_i / \xi_n) & \forall i:\ 1 \leqslant i < n \\ y_n = x_n / \xi_n \end{cases}$$

Now consider the mapping a from $\mathbf{R}^n$ to $\mathbf{R}^{n-1}$ defined by

$$\forall {}^t(x_1, \ldots, x_n) \in \mathbf{R}^n$$

$$(18) \quad a(x_1, \ldots, x_n) = \xi_n(y_1, \ldots, y_{n-1})$$

where y_i are given by (17). Obviously, the function a is linear. Moreover, since $\xi \in Z^n$, it maps Z^n on Z^{n-1}. From this mapping, we may build an LAF or a QLAF for (D,Θ) as described by the following theorem:

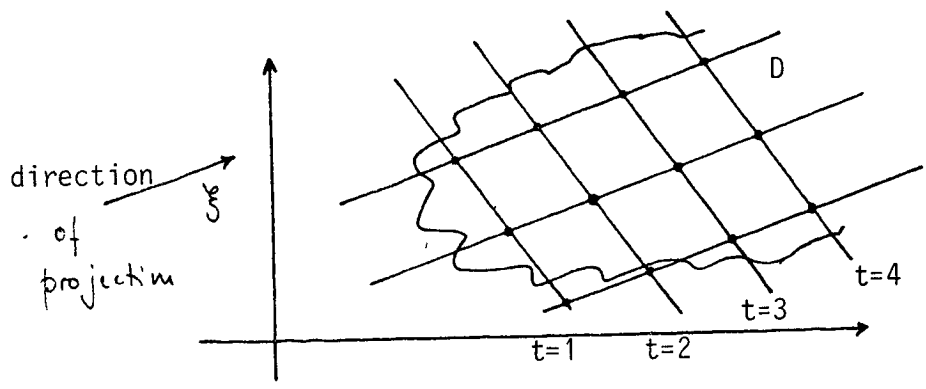


Fig. 7: Illustration of the principle of allocating a domain by a mapping which is conflict-free relative to the timing-function.

4. ALLOCATION FUNCTIONS

As explained in part 2, the third step of our method is to map the computation graph on a finite network of processors. Here also, we shall restrict ourselves to a particular class of mappings that we shall call **linear allocation functions** (LAF). Let n be the dimension of the domain D. Suppose $t=(\lambda,\alpha)$ is a QATF for (D,Θ). An allocation function is a mapping a from D to a finite subset of Z^m, where m is the dimension of the systolic array to be obtained. In the following, m will be taken as either equal to n-1 or to n, depending on t being an ATF or a strict QATF (this point will be made clear later on). However, it must be pointed out that the dimension of the systolic array may be chosen arbitrarily provided the allocation function ensures that the following condition is fullfilled

(14) $\forall x,y \in D \quad a(x)=a(y) \Rightarrow t(x) \neq t(y)$

This condition ensures that the allocation function is conflict-free relative to the timing function t.

An allocation function a from D to Z^m is said to be **linear** if a is a linear mapping from Z^n to Z^m. a is said to be **quasi-linear** (QLAF) if there exists a mapping a' from Z^n to Z^m and a vector P of Z^m such that

(15) $a(x) = a'(x) \bmod P$

We shall now distinguish two cases, namely when t is an ATF, and t is a strict QATF with rational coefficients.

4.1. t is an ATF

Let $t=(\lambda,\alpha(\lambda))$ be such a function. The idea is simply to project the dependence graph along a direction which is not parallel to the timing hyperplanes. We are thus ensured that condition (14) is fullfilled, since every projection line has at most one intersection point with a given timing hyperplane (see Fig.7). Depending on the domain D and on the projection direction that is chosen, one can obtain a finite machine or not. First of all, it should be clear that the domain D and the timing-function must be such that the number of points to be computed at a given time is bounded, i.e.

(16) $\exists M \in \mathbf{N}, \forall n \in \mathbf{N}, \mathrm{card}\{x \mid x \in D \text{ and } t(x)=n\} \leq M$

Such a condition holds clearly when D is finite. But in some other cases (for example, the convolution product), one can map an infinite domain D onto a finite machine. In the following, we shall suppose that D has at most one ray, say r.

Let ξ be a non-zero vector of Z^n. In the following, we shall suppose that ξ and the ray r of D are the smallest integer-coordinate vector in the direction they define (ξ and r are defined uniquely up to a non-zero scalar multiple). If this is the case, any line $L = \{x+\mu v \mid \mu \in \mathbf{R}\}$ (where v is ξ or r) is such that

$$Z^n \cap L = \{x+kv \mid k \in Z\}$$

Since ξ is non-zero, one of its coordinates is non-zero at least. For commodity, suppose $\xi_n \neq 0$. Let us denote e_i ($1 \leqslant i \leqslant n$) the canonical basis vectors of $\mathbf{R}^n$. It is clear that $\{e_1, \ldots, e_{n-1}, \xi\}$ is a basis of $\mathbf{R}^n$. Let $x = {}^t(x_1, \ldots, x_n)$ be a point of $\mathbf{R}^n$, and let $y = {}^t(y_1, \ldots, y_n)$ be the decomposition of x over this new basis. A simple calculation shows that the y_i satisfy the following relations:

$$(17) \quad \begin{cases} y_i = x_i - x_n (\xi_i / \xi_n) & \forall i: 1 \leqslant i < n \\ y_n = x_n / \xi_n \end{cases}$$

Now consider the mapping a from $\mathbf{R}^n$ to $\mathbf{R}^{n-1}$ defined by

$$\forall {}^t(x_1, \ldots, x_n) \in \mathbf{R}^n$$

$$(18) \quad a(x_1, \ldots, x_n) = \xi_n(y_1, \ldots, y_{n-1})$$

where y_i are given by (17). Obviously, the function a is linear. Moreover, since $\xi \in Z^n$, it maps Z^n on Z^{n-1}. From this mapping, we may build an LAF or a QLAF for (D,Θ) as described by the following theorem:

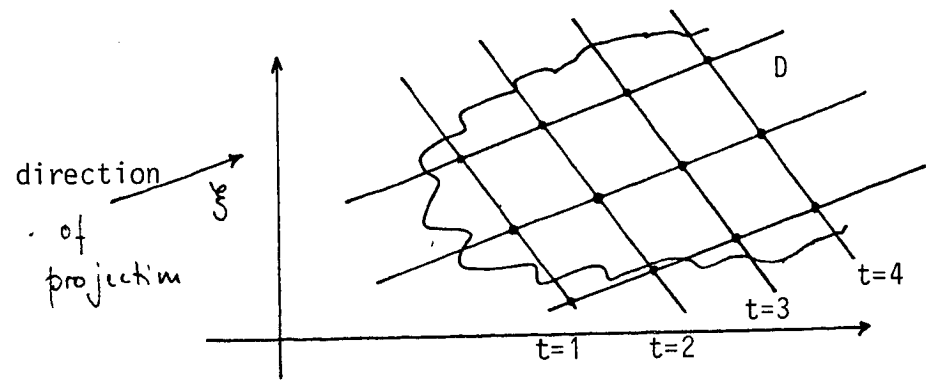


Fig. 7: Illustration of the principle of allocating a domain by a mapping which is conflict-free relative to the timing-function.

Theorem 5

Suppose that ${}^t\lambda r \neq 0$. Then

(i) If $\xi = r$, the mapping a defined by (18) is an LAF for (D,Θ) and t.

(ii) Suppose that ξ verifies ${}^t\lambda \xi \neq 0$, and that ξ is not collinear to r. Put

$$(19) \quad p = \left\lceil \left(\left| {}^t\lambda(M\xi) \right| + 1 \right) / {}^t\lambda r \right\rceil \quad \text{and} \quad P = p \; a(r)$$

where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x, and M is any integer such that $\forall x \in D, \forall n > M, x + n\xi \notin D$. Then the mapping:

$$a' : Z^n \dashrightarrow Z^{n-1}$$

$$x \mapsto a(x) \bmod P$$

is a QLAF for D.

Proof:

(i) According to the definition of a LAF, we have to prove that a(D) is finite and a is conflict-free relative to t. a(D) is finite since we supposed that ${}^t\lambda r \neq 0$. Consider two points z_1 and z_2 such that $z_1 \neq z_2$ and $a(z_1) = a(z_2)$. Necessarily, there exists some non-zero integer $k \in Z$ such that $z_2 = z_1 + k\xi$. Since t is an ATF, $t(z_2) = t(z_1) + k\,{}^t\lambda\xi$. Thus, $t(z_1) \neq t(z_2)$.

(ii) Notice first that M is finite, since we supposed that ξ is not collinear to r. That $a'(D)$ is finite is obvious. We have now to prove that a' is conflict-free relative to t. Let z_1 and z_2 be two points of D such that $z_1 \neq z_2$ and $a'(z_1) = a'(z_2)$. There exists a non-zero integer $k \in N$ such that $a(z_1) = a(z_2) + kP$. One can always suppose $k > 0$ by interchanging z_1 and z_2. We have then

$$a(z_1) = a(z_2) + k\,p\,a(r) = a(z_2 + k\,p\,r)$$

which implies that there exists some non-null $k' \in Z$ such that

$$z_1 = z_2 + k'\xi + k\,p\,r$$

Suppose now that $t(z_1) = t(z_2)$, i.e. ${}^t\lambda z_1 = {}^t\lambda z_2$. Then we would have

$$-k'\,{}^t\lambda\xi = k\,p\,{}^t\lambda r = k \left\lceil \left(\left| {}^t\lambda(M\xi) \right| + 1 \right) / {}^t\lambda r \right\rceil > k\,M \left| {}^t\lambda\xi \right|$$

from which

$$|k'| > k\,M$$

Since $k \neq 0$, we have

$$|k'| > kM \geqslant M$$

which is impossible. **QED**

4.2 t is a QATF

We suppose that t has rational coefficients (the case where $\lambda \in \mathbf{R}^n - \mathbf{Q}^n$ is uninteresting, since we deal with domains of $\mathbf{Z}^n$). We always suppose that D has at most one ray r.

Let ξ be a vector of $\mathbf{Z}^n$, and suppose that ${}^t\lambda\xi \neq 0$. Since t is a QATF, we are no longer ensured that a given line $\{x+k\xi \mid k \in \mathbf{N}\}$ meets a given timing-hyperplane at most once. However, the number of intersection points is bounded, which allows us to project D onto a finite subset of $\mathbf{Z}^n$, as is described by the following theorem:

Theorem 7

Suppose that ${}^t\lambda r \neq 0$, and let ξ be a vector of $\mathbf{Z}^n$ such that ${}^t\lambda\xi \neq 0$. Consider the integer p_n defined by

$$p_n = \lceil 1/|{}^t\lambda\xi| \rceil$$

and let b be the mapping from $\mathbf{R}^n$ to $\mathbf{R}^n$ defined by

$$b(x_1, \ldots, x_n) = (a(x) \bmod P, y_n \bmod p_n)$$

where a, P have been defined previously. Then b is a QLAF for D.

The proof of this theorem is very simple and is left to the reader.

5. SPECIFICATION OF A SYSTOLIC SOLUTION

In this part, we explain how the complete specification of a systolic architecture may be obtained from the system of recurrent equations, a timing-function t, and an allocation function. Let a(D) be the projection of the domain D. We suppose that a(D) is finite, and of course that a is conflict-free relative to t. What we have to define is:

- the structure of every cell of a(D)
- the communications between the cells
- the program that any cell has to compute.

Let

$$E(z)\begin{cases} u_1(z) = f(u_1(z+\theta_1), \ldots, u_p(z+\theta_p)) \\ \\ u_i(z) = u_i(z+\theta_i) \quad i=2, \ldots, m \end{cases}$$

be the URE associated with the point z. For the sake of simplicity, we suppose that f is the same in every point of D. In the following, we investigate successively the general structure of the systolic array and the structure and functionning of each cell.

5.1 General structure of the systolic array

The systolic array has one cell per point of a(D). Each cell has p input ports, one for each input argument u_i, which are denoted by $I(u_i)$. In the same manner, each cell has p output ports named $O(u_i)$.

Port $I(u_i)$ of cell π is connected to port $O(u_i)$ of processor $\pi+a(\theta_i)$. Moreover, if there exists a point z in the set $a^{-1}(\pi)$ such that $z+\theta_i \notin D$, port $I(u_i)$ is connected to the host-memory. Port $O(u_i)$ of cell π is connected to port $I(u_i)$ of cell $\pi-a(\theta_i)$ and also to the host-memory if $a^{-1}(\pi)$ contains a point z such that $z-\theta_i \notin D$.

Consider a communication link between port $I(u_j)$ of cell π and $O(u_i)$ of cell $\pi+a(\theta_i)$. There is a delay of $t(\theta_i)$ units of time associated with this link, i.e. any data available on $O(u_i)$ at time t is effectively used at time $t+t(\theta_i)$ by cell $\pi+a(\theta_i)$.

It may be easily checked that every cell has a bounded number of ports, and each port is connected at most to one other cell and the host-memory.

5.2 Structure of each cell

Let $(z_1, z_2, \ldots, z_n, \ldots)$ be the points of $a^{-1}(\pi)$ ordered by increasing values of t(z). Let $P(z_i)$ be the program to be executed for point z_i. Since we supposed that the URE is the same over D, programs that are to be executed on every cell are identical, except that perhaps some input values are to be taken from the host for some z_i, and also that some output values are to be sent to the host. But since the number of ports is finite, the set of different programs for a cell is finite and bounded . It may be shown that the sequence $P(z_i)$ is either finite (if D is finite) or is periodic.

The above considerations, although very informal, show that a systolic solution may be completely specified and automatically designed once t and a are found. In the following section, we shall show how our method may be applied to solve some well known problems.

6. APPLICATIONS

This part of the paper deals with showing how the above described method may be applied to well known problems. We successively investigate various systolic designs for convolution , and matrix product.

6.1. Convolution

There exist many solutions for this problem. In the following section, we show how classical systolic arrays and new ones may be derived.

In part 2, we gave a first uniform recurrent system for the convolution equations. Let us suppose now that values w_k are kept from node i-2,k instead of from node i-1,k as in the previous form. We then obtain the new URE:

$$(20)\quad \begin{cases} y(i,k) = y(i,k-1) + w(i-2,k)\ x(i-1,k-1) \\ x(i,k) = x(i-1,k-1) \\ w(i,k) = w(i-2,k) \end{cases}$$

The dependence graph (D,Θ) for (20) is given by (see Fig.8)

$$(21)\quad D = \{\ 0 \leqslant k \leqslant K\ ;\ 0 \leqslant i\ \}$$

and

$$(22)\quad \Theta = \{\ \theta_y=(0,-1)\ ,\ \theta_x=(-1,-1)\ ,\ \theta_w=(-2,0)\ \}$$

Theorem 3 shows that a sufficient condition for (λ,α) to be a QATF is

$$(23)\quad \begin{cases} \lambda_2 \geqslant 1 \\ \lambda_1 + \lambda_2 \geqslant 1 \\ 2\lambda_1 \geqslant 1 \end{cases}$$

and

$$(24)\quad \alpha \leqslant \inf\ \{\ {}^t\lambda s \mid s \in S\ \}$$

where $S = \{\ (0,0)\ ,\ (0,K)\ \}$ is the set of vertices of D. The timing-function t defined by

$$t(i,k) = \lfloor i/2 + k \rfloor$$

meets these conditions.

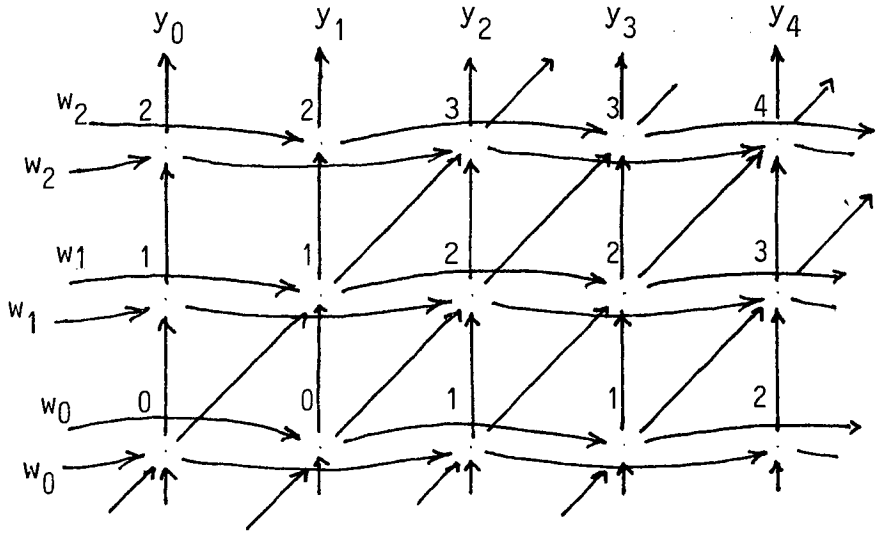


Fig. 8a: The dependence graph given by (20)

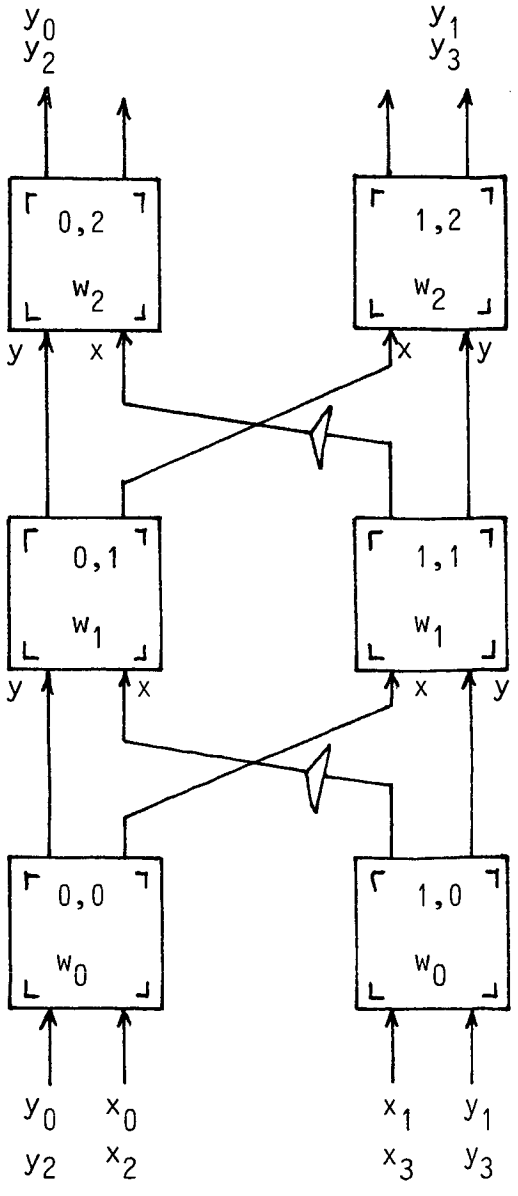


Fig. 8b: The **block-convolver.**

As shown by Fig.8, this QATF allows us to compute 2(K+1) values instead of (K+1) in the design of part 2. If we choose to project D along the i-axis, that is if we take the projection direction ξ=(1,0), we then obtain by applying theorem 6

(25) $\quad a(i,k) = (i,k) \bmod (2,0) = (i \bmod 2, k)$

which results in the systolic array depicted by Fig. 8. We call this design a **block-convolver** since values x_1 are input two at a time. Notice that this design may be extended without difficulty to the parallel computation of n (K + 1) values, by using a w- dependence $\theta_w = (-n,0)$.

Other solutions may be found by modifying the initial recurrent system of equations (2). For example, another way to serialize the computation of the y_i's is to proceed "backwards", i.e. by summing up terms involving w_k by decreasing values of k. One has then

$$\forall k: 0 \leqslant k \leqslant K\ ,\ \forall i: 0 \leqslant i$$

$$(26) \quad \begin{cases} y_{i,k} = y_{i,k+1} + w_k\, x_{i-k} \\ y_{i,K+1} = 0 \end{cases}$$

Final values of y_i are $y_{i,0}$. From (26), we may derive several URE, for example (see Fig. 9):

$$(27) \quad \begin{cases} y(i,k) = y(i,k+1) + w(i-1,k)\, x(i-1,k-1) \\ x(i,k) = x(i-1,k-1) \\ w(i,k) = w(i-1,k) \end{cases}$$

Applying theorem 3 gives the constraints over λ:

$$(28) \quad \begin{cases} -\lambda_2 \geqslant 1 \\ \lambda_1 \geqslant 1 \\ \lambda_1 + \lambda_2 \geqslant 1 \end{cases}$$

which are met by λ = (2,-1). α is then given by (24) and reaches its minimum value at the vertex (0,K) so that

$$t(i,k) = 2i - k + K = 2i + (K - k)$$

is a possible timing-function . By projecting D along the i-axis, we get the LAF a(i,k) = k which gives the well-known design of Fig.9.

Consider now the projection defined by ξ = (1,-1). Applying theorem 5 gives (for K=2) the QLAF

$$a(i,k) = (i + k) \bmod 4$$

Fig. 10 shows the systolic structure which is obtained once the cells are arranged on a ring. This design, called a **ring convolver**, has 4 cells. Cell i reads values y from cell (i+1 mod 4) or from the host. It reads values x from cell (i-2 mod 4) or from the host. It reads w from cell (i-1 mod 4). It sends y either to cell (i-1 mod 4) or to the host, and x either to cell (i+2 mod 4) or to the host. Finally, notice that w-connections between cell i and cell (i+1 mod 4) has a delay of two units of time. Each cell executes indefinitely a sequence of eight cycles P_1, Q, Q, P_2, Q, Q, P_3, Q, where Q denotes the idle cycle and P_1, P_2, P_3 are defined by

P_1: read y from the host; read w from cell (i-1 mod 4); read x from cell (i-2 mod 4)
send y + w x to cell (i-1 mod 4); send w to cell (i+1 mod 4); send x to cell (i+2 mod 4);

P_2: read y from cell (i+1 mod 4); read w from cell (i-1 mod 4); read x from cell (i-2 mod 4);
send y + w x to cell (i-1 mod 4); send w to cell (i+1 mod 4); send x to cell (i+2 mod 4);

P_3: read y from cell (i+1 mod 4); read w from cell (i-1 mod 4); read x from the host;
send y + w x to the host; send w to cell (i+1 mod 4); send x to cell (i+2 mod 4);

Depending on their number, the cells execute this sequence starting at one particular cycle. Table 11 shows the first ten cycles for the ring.

Although the above described designs are not equally interesting, they show that our method is very powerful and allows us to find automatically new systolic arrays for well known problems.

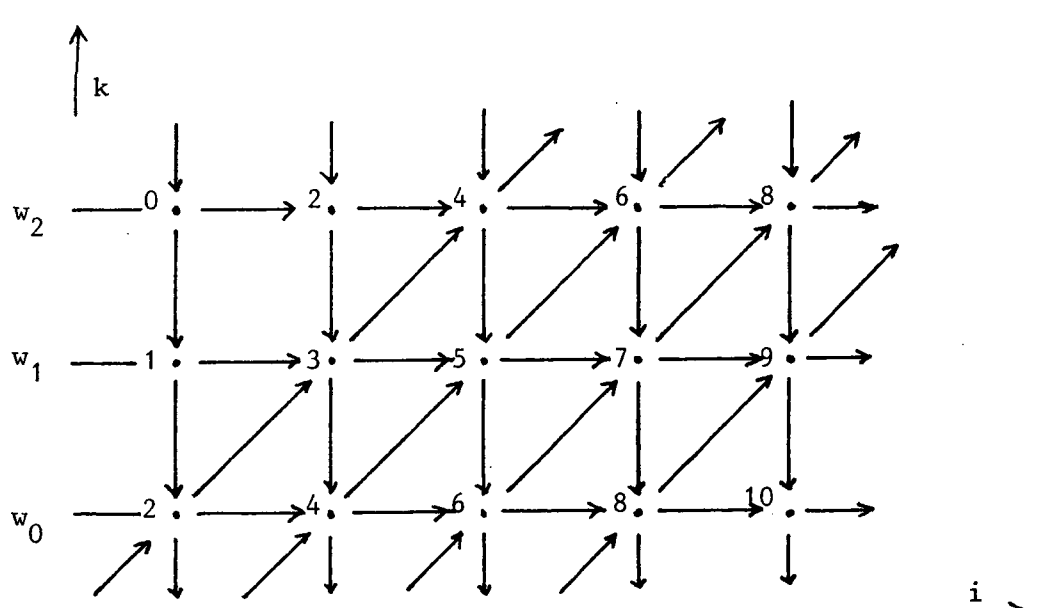


Fig. 9a: Dependence graph for the URE given by (26),

and timing-function t(i,k) = 2i + (K - k) for K = 2.

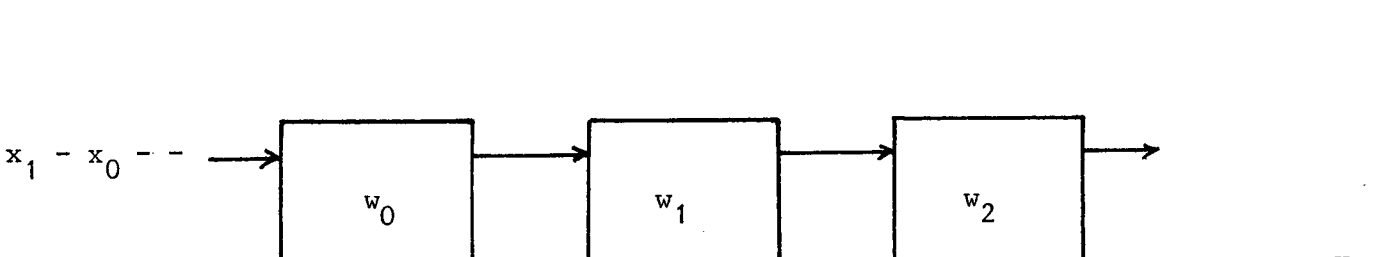


Fig. 9b: Systolic array for the dependence graph of Fig. 9a, when projected along the i-axis. Values that enter this array are shown from time t = 0. - denotes no value. (For example, x_0 enters cell 0 at t= , x_1 at time t=4 and so forth).

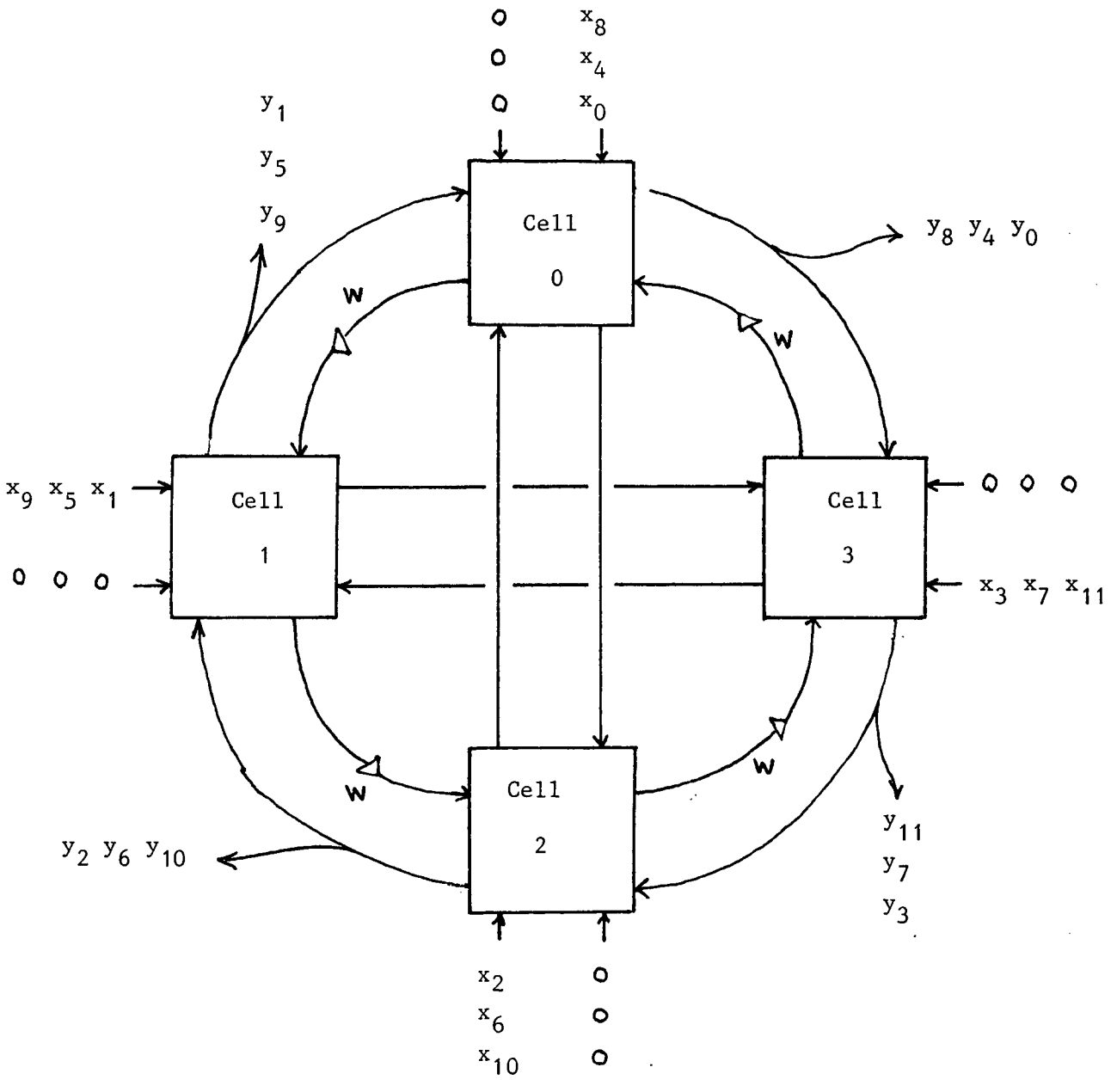


Fig. 10: The **ring-convolver** for K=2

Cycle number	Cell number 0	1	2	3
0	-	-	P_1	-
1	-	P_2	Q	-
2	P_3	Q	Q	P_1
3	Q	Q	P_2	Q
4	P_1	P_3	Q	Q
5	Q	Q	Q	P_2
6	Q	P_1	P_3	Q
7	P_2	Q	Q	Q
8	Q	Q	P_1	P_3
9	Q	P_2	Q	Q
10	P_3	Q	Q	P_1
	⋮	⋮	⋮	⋮

Table 11: Activity of the cells of the ring-convolver during the first ten cycles

6.2 Matrix product

Consider two N x N matrices A and B and let C = A x B. We have

$$(29) \qquad c_{i,j} = \sum_{k=1}^{N} a_{i,k}\, b_{k,j} \qquad 1 \leqslant i \leqslant N,\ 1 \leqslant j \leqslant N$$

As in the case of the convolution product, a natural way to serialize (29) is to put

$$\forall_{i,j,k}:\ 1 \leqslant i \leqslant N,\ 1 \leqslant j \leqslant N,\ 1 \leqslant k \leqslant N$$

$$(30) \qquad \begin{cases} c_{i,j,k} = c_{i,j,k-1} + a_{i,k}\, b_{k,j} \\ \\ c_{i,j,0} = 0 \end{cases}$$

It is clear that $a_{i,k}$ is used by nodes (i,j,k) where $1 \leqslant j \leqslant N$ and $b_{k,j}$ is used by nodes (i,j,k) where $1 \leqslant i \leqslant N$. A natural mode of circulation of these data is that node (i,j,k) takes $a_{i,k}$ from node (i,j-1,k) and $b_{k,j}$ from node (i-1,j,k). This leads to the following URE

$$(31) \qquad \begin{cases} c(i,j,k) = c(i,j,k-1) + a(i,j-1,k)\ b(i-1,j,k) \\ \\ a(i,j,k) = a(i,j-1,k) \\ \\ b(i,j,k) = b(i-1,j,k) \end{cases}$$

The dependence graph for (31) is (D,Θ) where

$$D = \{\ (i,j,k)\ |\ 1 \leqslant i \leqslant N,\ 1 \leqslant j \leqslant N,\ 1 \leqslant k \leqslant N \}$$

is a cube of $\mathbb{R}^n$ and

$$\Theta = \{\ \theta_c = (0,0,-1),\ \theta_a = (0,-1,0),\ \theta_b = (-1,0,0)\ \}.$$

Applying theorem 3 gives the following constraints over $\lambda = (\lambda_1, \lambda_2, \lambda_3)$

$$(35) \qquad \begin{cases} \lambda_1 \geqslant 1 \\ \\ \lambda_2 \geqslant 1 \\ \\ \lambda_3 \geqslant 1 \end{cases}$$

Thus t(i,j,k) = i + j + k - 3 is a possible solution. From D and t, we can derive three well known designs for matrix-product:

- Take ξ = (0,0,1), i.e. project D along axis k. We then get the design of Fig. 12;
- Take ξ = (1,1,0). We obtain the design of Fig.13, which has Nx(2N - 1) cells.
- Finally, with ξ = (1,1,1), we have the design described by KUNG ([4]), shown by Fig. 14.

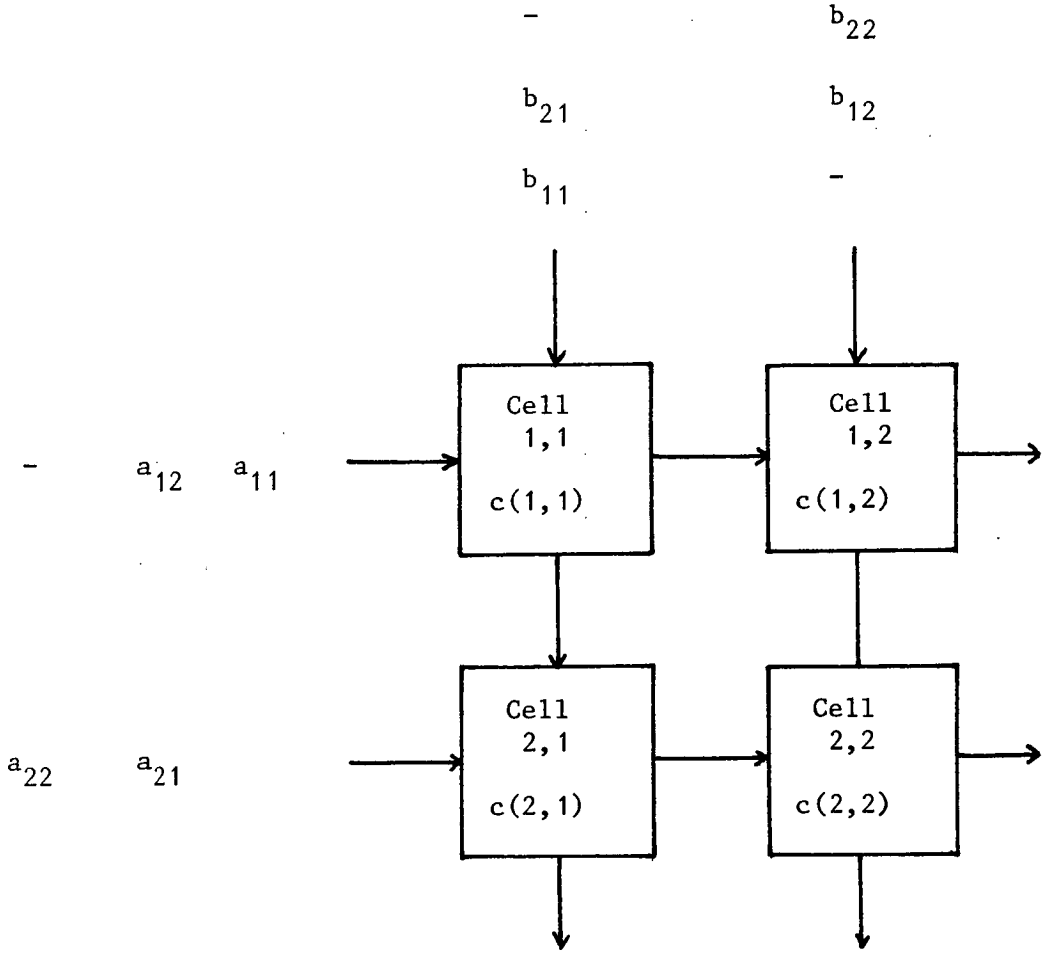
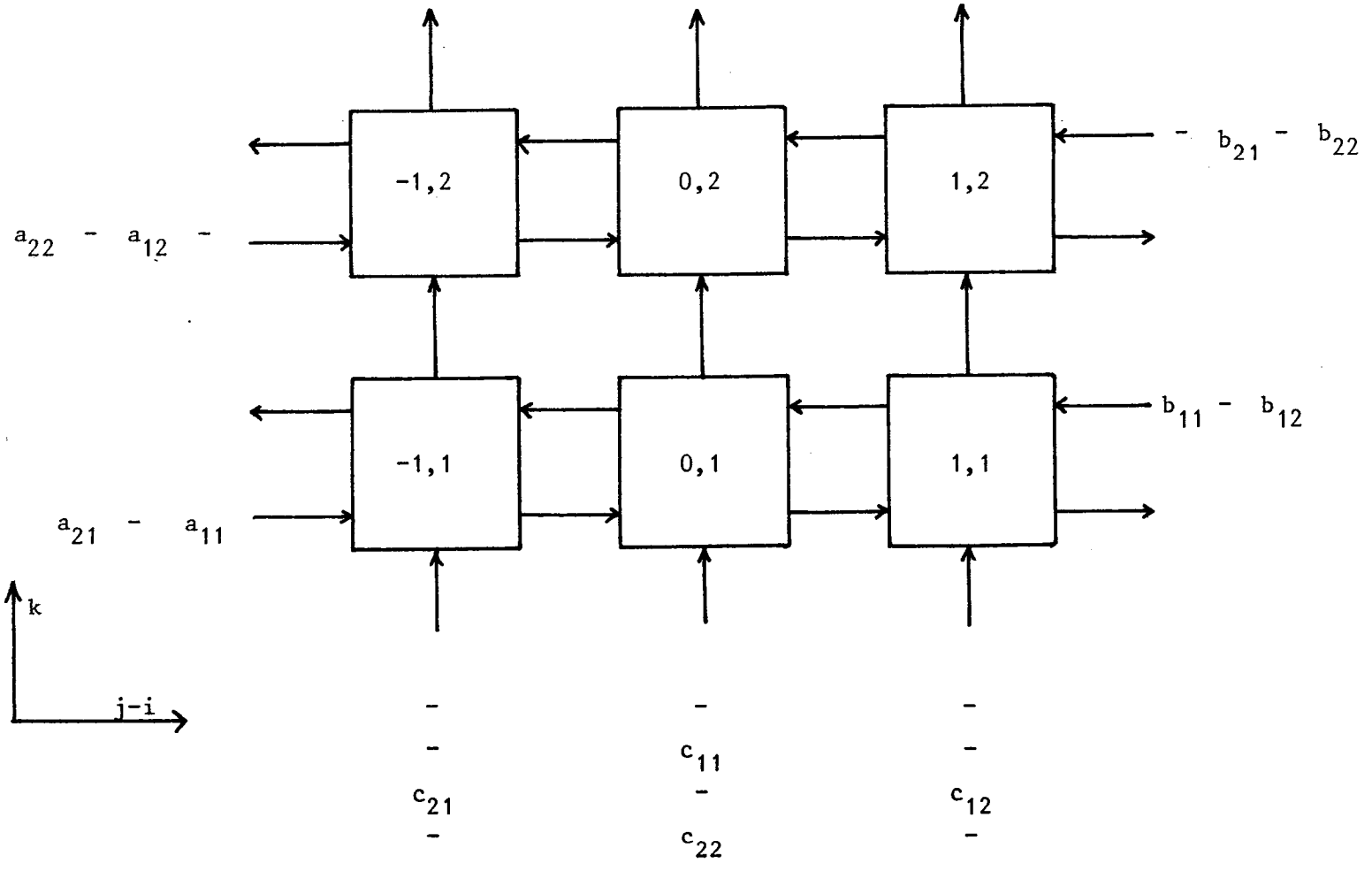


Fig. 12: First design for the matrix product; $c_{i,j}$ stay on cell (i,j). The network contains N^2 cells

Fig. 13: Second design for the matrix product with $O(2N^2)$ cells. The allocation function is $a(i,k) = (j-i,k)$



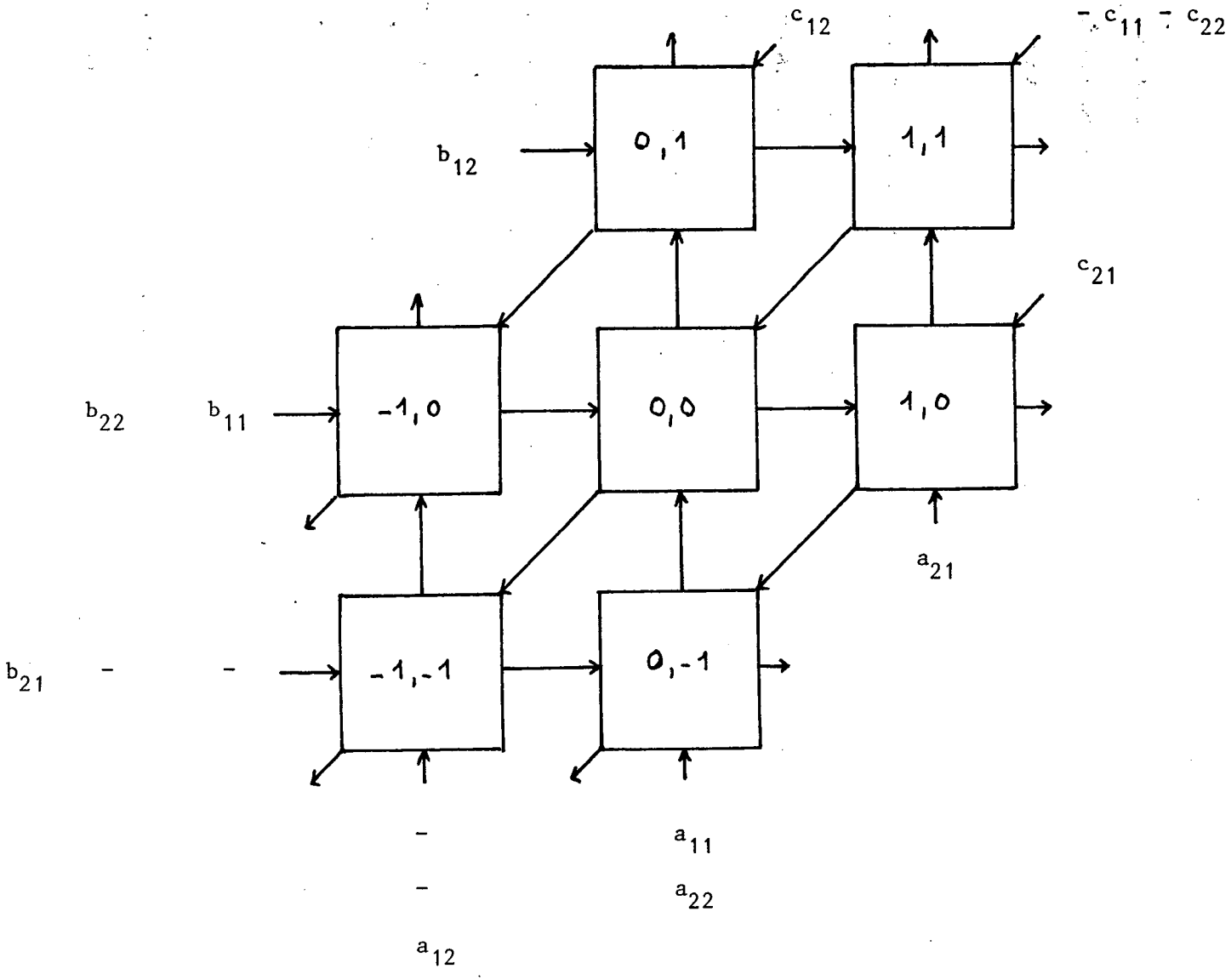


Fig.14: Third design for the matrix product with $O(3N^2)$ cells. The allocation function a(i,j,k) = (i-k,j-k) is obtained by a projection along ξ=(1,1,1)

7. CONCLUSION

In this paper, we described a new powerful method for the systematic design of systolic arrays. The method consists in deriving uniform recurrence equations for a given problem, then in finding a timing-function for the computations, and finally in projecting the domain of computation along a direction which is not parallel to the hyperplanes defined by the timing-function. Theorems that defines the condition upon which a timing-function and an allocation function exist have been given. These theorems are constructive and allow to build step by step a systolic array that implements a given algorithm, provided some simple conditions are met. Non trivial applications of this method have been described, that show that it can be used for finding entirely new designs.

The above described method is currently used as a basis for the design of a Computer Aided Design System for systolic arrays [4]. This CAD system, named DIASTOL, is intended to provide a designer with interactive graphics facilities and various simulation tools, which, to our sense, are necessary for a fast and reliable design of special-purpose architectures. A very interesting characteristic of DIASTOL is that the designs that are obtained in such a way are necessary correct and consequently do not need to be proven, as soon as the uniform recurrent system of equations has been proven to be formally equivalent to the mathematical equations of the problem.

Other attemps to synthesize systolic arrays have been recently proposed by MOLDOVAN ([5]) and MIRANKER and WINKLER ([6]). These two approaches are very similar; they both start from a program which defines the problem to be solved. The dependences between the variables of the program, considered as points of Z^n, are extracted, and both methods look for a linear transformation that maps these points onto an a priori given systolic structure. The main differences between these two approaches and ours are the following:

- our method starts from a system of recurrent equations, instead of from a program. Although it may be found in some cases more natural to express the initial problem as a program, we believe URE's offer more possibilities to be formally transformed and manipulated. Also, expressing a problem as a program implicitly supposes that we have at least one sequential execution scheme for the problem. In our approach, this hypothesis is not made, and the conditions under which an execution scheme exists are given in the theorems that have been given.

- another very important difference lies in the fact that our method is constructive; the systolic structure results completely from the method, and thus, we have a tool that allows to investigate systematically a large set of possible systolic arrays for a given algorithm.

Among the numerous open problems that may be investigated, let us cite only two that seem fundamental. The first one is to explore formally which class of algorithms may be expressed by means of URE, and to define formal transformations that allow to derive equivalent URE's. Another problem to be solved is the evaluation of the solutions that are obtained by such a method. This is a very difficult problem, since a quality measure must include a number of parameters such as size, speed, memory-bandwith, which need to be evaluated not only formally, but also relative to the function that a systolic array has to perform, i.e., the system in which it is to be included.

Aknowledgments

I would like to aknowledge many people for their help during the elaboration of the above reported work. The idea and proof of theorem 2 are due to my brother Pascal QUINTON, who also contributed greatly to the formal explanation of many informal ideas. Thanks also to Philippe DARONDEAU, with whom I had a number of discussions about this work. Finally, Françoise ANDRE, Albert BENVENISTE and Laurent KOTT read this paper very carefully and correct a number of errors.

REFERENCES

[1] KARP R.M. , MILLER R.E. , WINOGRAG S. , The Organization of Computations for Uniform Recurrence Equations, JACM, Vol. 14, No 3, July 1967, pp. 563-590.

[2] KUNG H.T. , Why Systolic Architectures? , Computer, Vol. 15, No 1, Jan. 1982, pp. 37-46

[3] MEAD C. , CONWAY L. , Introduction to VLSI Systems, Chapter 8 by KUNG H.T. , Addison Wesley, 1980.

[4] ANDRE F. , FRISON P. , QUINTON P. , DIASTOL: un système de conception assistée pour les architectures systoliques, IRISA Internal Report, to appear.

[5] MOLDOVAN D.I. , On the Design of Algorithms for VLSI Systolic Arrays, Proceedings of the IEEE, Vol. 71, No. 1, Jan. 1983, pp. 113-120.

[6] MIRANKER W.L. , WINKLER A. , Spacetime Representations of Systolic Computational Structures, IBM Research Report RC 9775, Dec. 1982.

[7] ROCKAFELLAR R.T., Convex Analysis, Princeton University Press, 1970.

Liste des Publications Internes IRISA

PI 150 **Construction automatique et évaluation d'un graphe d'«implication» issu de données binaires, dans le cadre de la didactique des mathématiques**
H. Rostam , 112 pages ; Juin 1981

PI 151 **Réalisation d'un outil d'évaluation de mécanismes de détection de pannes]-]Projet Pilote SURF**
B. Decouty, G. Michel, C. Wagner, Y. Crouzet , 59 pages ; Juillet 1981

PI 152 **Règle maximale**
J. Pellaumail , 18 pages ; Septembre 1981

PI 153 **Corrélation partielle dans le cas « qualitatif »**
I.C. Lerman , 125 pages ; Octobre 1981

PI 154 **Stability analysis of adaptively controlled not-necessarily minimum phase systems with disturbances**
Cl. Samson , 40 pages ; Octobre 1981

PI 155 **Analyses d'opinions d'instituteurs à l'égard de l'appropriation des nombres naturels par les élèves de cycle préparatoire**
R. Gras , 37 pages ; Octobre 1981

PI 156 **Récursion induction principle revisited**
G. Boudol, L. Kott , 49 pages ; Décembre 1981

PI 157 **Loi d'une variable aléatoire à valeur R^+ réalisant le minimum des moments d'ordre supérieur à deux lorsque les deux premiers sont fixés**
M.Kowalowka, R. Marie , 8 pages ; Décembre 1981

PI 158 **Réalisations stochastiques de signaux non stationnaires, et identification sur un seul échantillon**
A. Benveniste J.J. Fuchs , 33 pages ; Mars 1982

PI 159 **Méthode d'interprétation d'une classification hiérarchique d'attributs-modalites pour l'«explication» d'une variable ; application à la recherche de seuil critique de la tension artérielle systolique et des indicateurs de risque cardiovasculaire**
B. Tallur , 34 pages ; Janvier 1982

PI 160 **Probabilite stationnaire d'un réseau de files d'attente multiclasse à serveur central et à routages dépendant de l'état**
L.M. Le Ny , 18 pages ; Janvier 1982

PI 161 **Détection séqentielle de changements brusques des caractéristiques spectrales d'un signal numérique**
M. Basseville, A. Benveniste , pages ; Mars 1982

PI 162 **Actes regroupés des journées de Classification de Toulouse (Mai 1980), et de Nancy (Juin 1981)**
I.C. Lerman , 304 pages ;

PI 163 **Modélisation et Identification des caractéristiques d'une structure vibratoire : un problème de réalisation stochastique d'un grand système non stationnaire**
M. Prévosto, A. Benveniste, B. Barnouin , 46 pages ; Mars 182

PI 164 **An enlarged definition and complete axiomatization of observational congruence of finite processes**
Ph. Darondeau , 45 pages ; Avril 1982

PI 165 **Accès vidéotex à une banque de données médicales**
A. Chauffaut, M. Dragone, R. Rivoire, J.M. Roger , 25 pages ; Mai 1982

PI 166 **Comparaison de groupes de variables définies sur le même ensemble d'individus**
B. Escofier, J. Pages , 115 pages ; Mai 1982

PI 167 **Transport en circuits virtuels internes sur réseau local et connexion Transpac**
M. Tournois, R. Trépos , 90 pages ; Mai 1982

PI 168 **Impact de l'intégration sur le traitement automatique de la parole**
P. Quinton , 14 pages ; Mai 1982

PI 169 **A systolic algorithm for connected word recognition**
J.P. Banâtre, P. Frison, P. Quinton , 13 pages ; Mai 1982

PI 170 **A network for the detection of words in continuous speech**
J.P. Banâtre, P. Frison, P. Quinton , 24 pages ; Mai 1982

PI 171 **Le langage ADA : Etude bibliographique**
J. André, Y. Jégou, M. Raynal , 12 pages ; Juin 1982

PI 172 **Comparaison de groupes de variables : 2ème partie : un exemple d'application**
B. Escofier, J. Pajès , 37 pages ; Juillet 1982

PI 173 **Unfold-fold program transformations**
L. Kott , 29pages ; Juillet 1982

PI 174 **Remarques sur les langages de parenthèses**
J.M. Autebert, J. Beauquier, L. Boasson, G. Senizergues , 20 pages ; Juillet 1982

PI 175 **Langages de parenthèses, langages N.T.S. et homomorphismes inverses**
J.M. Autebert, L. Boasson, G. Senizergues , 26 pages ; Juillet 1982

PI 176 **Tris pour machines synchrones ou Baudet Stevenson revisited**
R. Rannou , 26 pages ; Juillet 1982

PI 177 **Un nouvel algorithme de classification hiérarchique des éléments constitutifs de tableau de contingence basé sur la corrélation**
B. Tallur , Juillet 1982 ;

PI 178 **Programmes d'analyse des re'sultats d'une classification automatique**
I.C. Lerman et collaborateurs , 79 pages ; Septembre 1982

PI 179 **Attitude à l'égard des mathématiques des élèves de sixième**
J.Degouys, R. Gras, M. Postic , 29 pages ; Septembre 1982

PI 180 **Traitements de textes et manipulations de documents : bibliographie analytique**
J. André , 20 pages ; Septembre 1982

PI 181 **Algorithme assurant l'insertion dynamique d'un processeur autour d'un réseau à diffusion et garantissant la cohérence d'un système de numérotation des paquets global et réparti**
Annick Le Coz, Hervé Le Goff, Michel Ollivier . 31 pages ; Octobre 1982

PI 182 **Interprétation non linéaire d'un coefficient d'association entre modalités d'une juxtaposition de tables de contingence**
Israël César Lerman . 34 pages ; Novembre 1982

PI 183 **L'IRISA vu à travers les stages effectués par ses étudiants de DEA (1è année de thèse)**
Daniel Herman . 41 pages ; Novembre 1982

PI 184 **Commande non linéaire robuste des robots manipulateurs**
Claude Samson . 52 pages ; Janvier 1983

PI 185 **Dialogue et représentation des informations dans un système de messagerie intelligent**
Philippe Besnard, René Quiniou, Patrice Quinton, Patrick Saint-Dizier, Jacques Siroux, Laurent Trilling . 45 pages ; Janvier 1983

PI 186 **Analyse classificatoire d'une correspondance multiple ; typologie et régression**
I.C. Lerman . 54 pages ; Janvier 1983

PI 187 **Estimation de mouvement dans une sequence d'images de télévision en vue d'un codage avec compensation de mouvement**
Claude Labit . 132 pages ; Janvier 1983

PI 188 **Conception et réalisation d'un logiciel de saisie et restitution de cartes élémentaires**
Eric Sécher . 45 pages ; Janvier 1983

PI 189 **Etude comparative d'algorithmes pour l'amélioration de dessins au trait sur surfaces point par point**
M.A. ROY . 96 pages ; Janvier 1983

PI 190 **Généralisation de l'analyse des correspondances à la comparaison de tableaux de fréquence**
Brigitte Escofier . 35 pages ; Mars 1983

PI 191 **Association entre variables qualitatives ordinales «nettes» ou «floues»**
Israël-César Lerman . 42 pages ; Mars 1983

PI 192 **Un processeur intégré pour la reconnaissance de la parole**
Patrice Frison . 80 pages ; Mars 1983

PI 193 **The Systematic Design of Systolic Arrays**
Patrice Quinton . 39 pages ; Avril 1983

PI 194 **Régime stationnaire pour une file M/H/1 avec impatience**
Raymond Marie et Jean Pellaumail . 8 pages ; Mars 1983

PI 195 **SIGNAL : un langage pour le traitement du signal**
Paul Le Guernic, Albert Benveniste, Thierry Gautier . 49 pages ; Mars 1983

PI 196 **Algorithmes systoliques : de la théorie à la pratique**
Françoise André, Patrice Frison, Patrice Quinton . 19 pages ; Mars 1983

PI 197 **HAVANE : un système de mise en relation automatique de petites annonces**
Patrick Bosc, Michèle Courant, Sophie Robin, Laurent Trilling . 79 pages ; Mai 1983

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique